published as: W. Zhou, V. H. Raja, Christoph Meinel: An Authentication and Authorization System for Virtual Organizations;
Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI 2005);
Florida, USA, July 2005, Vol. VII, pp. 150-155.

An Authentication and Authorization System for Virtual Organizations

Wei ZHOU, Vinesh H. RAJA
School of Engineering, University of Warwick
Coventry CV4 7AL, UK

Christoph MEINEL
Hasso Plattner Institute, University of Potsdam
Potsdam D-14440, Germany

ABSTRACT

As more businesses engage in globalization, inter-organizational collaborative computing grows in importance. In order to effectively participate in modern collaborations, member organizations must be able to share specific data and functionality with collaboration partners, while ensuring that their resources are safe from inappropriate access. This requires access control models, policies, and enforcement mechanisms for collaboration resources. This paper describes a mechanism that can exchange user's authentication and authorization information between independent organizations in a secured way, and how this mechanism can be used to carry out the privilege management for virtual organization. The basic principle is that a user is authenticated locally at his origin site, and the origin site creates a handle to be used to retrieve attributes about the user for the resource provider. According to the user's handle, the resource provider sends an attribute request to the user's attribute authority. The attribute authority then issues a X.509 attribute certificate that holds the user's attributes and sends it back to the requester. The resource provider then provides required services to the user based on his privileges. A prototype called Cross Security Access Control Framework (CSACF) has been developed.

Keywords: Authentication, Authorization, Role-based Access Control, Privilege Management Infrastructure, Attribute Certificates, Virtual Organization.

1. INTRODUCTION

With the advent of the information superhighway, businesses, governments and other organizations co-operate in innovative ways. To effectively participate in modern collaborations, member organizations must be able to share specific data and functionality with collaboration partners, while ensuring that their resources are safe from inappropriate access. Such collaborations may dynamically change participants and trust relationships during the life cycle, and each organization may join in several inter-organizational collaborations. This requires access control models, policies, and enforcement mechanisms for shared resources. Unfortunately, current technologies do not comprehensively support such control for collaboration resource sharing. Though there are a variety of conceptual, technological and operational factors that contribute to this situation, we specifically address the following problem: how to exchange authentication and authorization information in the inter-organizational collaborative computing environment, and how this mechanism can be used for supporting the authorization management in virtual organizations.

In this paper, we propose a mechanism for exchanging authentication and authorization information between independent organizations in a secured way. The basic idea is that a user is authenticated locally at his origin site (identity provider), and the origin site creates a handle that is used to retrieve attributes (privileges) about the user. This handle is stored in a XML document. This XML document is signed and encrypted before it is sent to the destination side (resource provider). According to the user's handle, the resource provider sends an attribute request to the user's attribute authority for his authorization information. The attribute authority will issue a X.509 attribute certificate that holds the user's privileges and send it back to the resource provider. The resource provider verifies the attribute certificate and extracts the user's privileges from it, and then provides required services to the user based on his privileges.

The rest of this paper is organized as follow. Section 2 introduces the basic concepts of privilege management infrastructure, XML Signature and XML Encryption. Section 3 introduces our Cross Security Access Control Framework (CSACF). Section 4 describes a role mapping mechanism between organizations. Section 5 describes how the CSACF supports virtual organization management. Section 6 describes implementation methodology. Section 7 compares our work to some related works. Finally, section 8 gives the conclusions and future works.

2. REALATED TECHNOLOGIES INTRODUCTION

Privilege management infrastructure

Privilege Management Infrastructure (PMI) is specified by the ITU-T and ISO/IEC [3]. The main function of PMI is to provide a strong authorization after the authentication has taken place. It has a number of similarities with Public Key Infrastructure (PKI) [4]. The basic data structure in PMI is X.509 Attribute Certificate (AC) [5]. Like Public Key Certificate (PKC) strongly binds a public key to its subject, AC strongly binds a set of attributes to its holder. In fact, attribute certificates have been designed to be used in conjunction with identity certificates, i.e. PMI and PKI are linked by information contained in the ACs and PKCs. For example the holder field in an AC contains the serial number and issuer of a PKC. The attribute certificate, identity certificate and their relation are depicted in Figure 1.

In a PMI, the entity that digitally signs an AC is called an Attribute Authority (AA). The trusted root of a PMI is called Source of Authority (SOA). A SOA may delegate its powers of

authorization to subordinate AAs. Subordinate AAs may also delegate their powers of authorization to further subordinate AAs. Then an AA hierarchy can be established. When a user's authorization permissions need to be revoked, an AA will issue an Attribute Certificate Revocation List (ACRL) containing the list of ACs that no longer to be trusted.

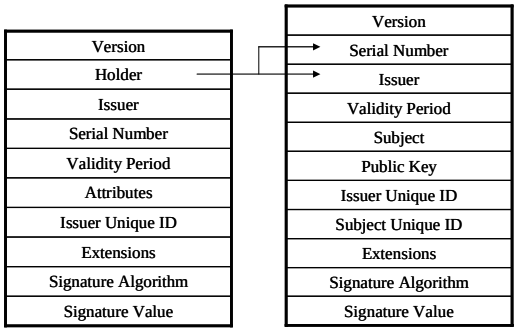

Figure 1: Relation between attribute and identity certificate

There are two primary models for distribution of ACs: the "push" and "pull" model. In "push" model, the client supplies his AC to a server at the time of request. The "push" model is suitable when the client's rights should be assigned within the client's " home" domain. In the "pull" model, the server retrieves the client's AC from an AC repository. The "pull" model is suitable when the client's rights should be assigned within the server's domain. The choice to use the "push" or "pull" method is dependent on system requirements and the available infrastructure.

ACs may be used with various security services, including access control, data origin authentication, and non-repudiation. In our work we use ACs to store the authorization information, e.g. users' roles and access control policies.

XML Signature and XML Encryption

Both XML Signature [6] and XML Encryption [7] are W3C proposed recommendations. XML Signature provides syntax for representing signatures on digital content along with procedures for computing and verifying such signatures. XML Signatures can be applied to any kind of data of any format, including XML document. XML Signature lets a user sign specific portions of the XML tree rather than the complete document. The XML Signature standard also allows a user to filter and transform the data before he signs it and lets him choose exactly what to sign and how.

XML Encryption specifies a format and processing for encrypting data in XML. The data may be arbitrary data, including XML document. With XML Encryption, there is a lot of flexibility in how documents are encrypted. For instance, different nodes of an XML document could be encrypted with different keys, while some nodes are left in plain text. With XML Encryption, we can manage situations where different parts of the same document need different treatment, i.e. encrypting part of the data being exchanged.

The XML Signature and XML Encryption standards are being used extensively as building-block technologies. We use XML Signatures and XML Encryption to sign and encrypt the partial or whole XML documents, e.g. the SOAP messages.

3. CROSS SECURITY ACCESS CONTROL FRAMEWORK

Cross security access control framework

The Cross Security Access Control Framework (CSACF) is a framework that provides cross-organizational access control. Its architecture as depicted in Figure 2 is composed of two independent parts: Access Control Engine (ACE) and Credential Service Centre (CSC). The ACE is responsible for access control. The basic components of ACE are Access control Enforcement Function (AEF) and Access control Decision Function (ADF). The CSC is responsible for authentication and obtaining users' attributes that are used for access control. The basic components of CSC are Authentication Service (AuthS) and Attribute Service (AttrS).

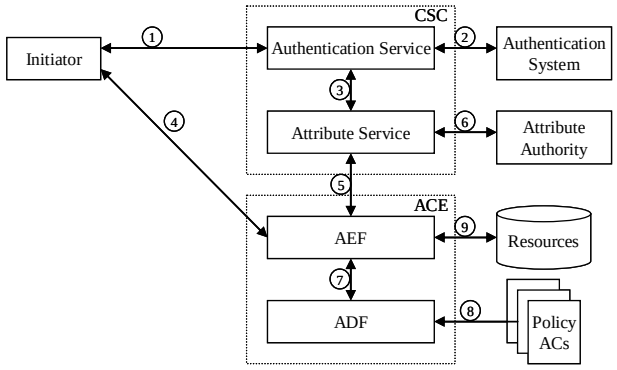

Figure 2: Cross security access control framework

Access control engine

Role-based access control (RBAC) emerged rapidly in the 1990s as a proven technology for managing and enforcing security in large-scale enterprise wide systems [1, 2]. In RBAC, access rights are associated with roles, and users are assigned appropriate roles thereby acquiring the corresponding permissions. This approach can provide more flexibility to security management over the traditional approach of using user and group identifiers.

We have developed a RBAC system with X.509 attribute certificates. The ACs used in the system can be classified into two categories; namely role ACs that store users' roles and policy ACs that store authorization policies. The authorization policies specify which roles have what rights on various targets. All the access control decisions are made based on the authorization policies. Role and policy ACs are stored in LDAP servers. The heart of the system is an access control engine (ACE). The ACE executes the functions of authentication and authorization, and then accesses the targets on behalf of the user.

Our access control framework conforms to the basic principle of ISO 10181-3 Access Control Framework that is defined by the Open Group [8]. This framework separates authentication from authorization, and comprises of four components: Initiator (e.g. a user), Target (e.g. a database), Access control Enforcement Function (AEF) and Access control Decision Function (ADF). After passing authentication, the initiator submits access request that specifies an operation to be performed on a target. The AEF mediates access request, it submits decision requests to ADF. ADF decides whether access requests should be granted or denied based on the user's roles and access control policies.

Finally, AEF enforces access control decisions made by ADF. More information about the ACE can be referred from [9].

Authentication service

Authentication Service (AuthS) performs the functions related to users' authentication. It can accomplish three tasks. The first is redirecting a user to his origin site for authentication. The second is connecting to local authentication system so that the user is authenticated at his origin site, and then creating a handle that is used to retrieve attributes about the user. The third is forwarding the user's handle back to the destination site and performing impersonation check to the received handle. There may be multi-CSC between the destination site and the origin site. An institution must register to a CSC in order to get its service. The functional components of AuthS are depicted in Figure 3 and described below.

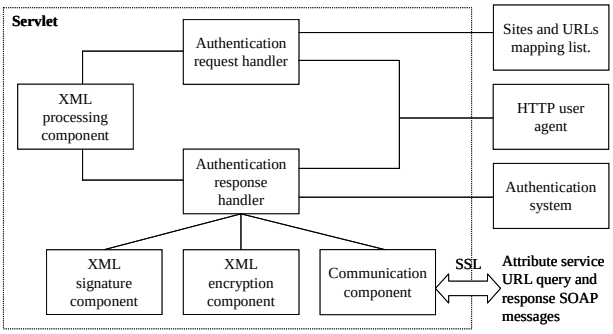

Figure 3. The authentication service structure

- *Authentication request handler* performs two primary functions. One is interacting with a user and finding out his origin site. The other is redirecting the user's browser to his origin site. The sites' location information is stored in a "Sites and URLs mapping list". After getting the user's origin site URL, the authentication request handler creates an authentication request message, and redirects the user's browser to that URL. The authentication request message is a XML document that describes the authentication response acceptance URL and the user's desired target URL. The authentication request is originally created by an ACE, and contains the target information, i.e. the URL that the user wants to visit. The new authentication request is created through adding new information to the old message received from an ACE or another AuthS.
- *Authentication response handler* performs four primary functions. The first is interacting with a user to get him "logged in" to origin site's authentication system. The second is creating a handle that is used to retrieve the user's attributes. If the AuthS does not have enough information to create a handle, it will contact the AttrS for it. The third is performing impersonation check to all the received handles. The fourth is forwarding a user's handles back to the destination site, i.e. another AuthS or the user's desired target URL. The user's handle is encapsulated in a handle response message that is XML document. This message is signed and then encrypted. The impersonation check is done through decrypting and verifying the XML document.
- *XML processing component* performs the functions related to creating authentication request message and handle response message.
- *XML signature component* is responsible for signing and verifying XML documents.
- *XML encryption component* is responsible for encrypting and decrypting XML documents.
- *Communication component* is responsible for communicating with AttrS to get a user's attribute query URL. The connection is made through mutual authentication over SSL.

Attribute service

Attribute Service (AttrS) performs the functions related to retrieving users' attributes. It can accomplish three major tasks. The first is interacting with attribute authority to get attributes about a user, and then mapping these attributes from one format to another format if it is required. The second is issuing an X.509 attribute certificate that holds the user's attributes. The third is forwarding the user's AC back to the requester. The functional components of AttrS are depicted in Figure 4 and described below.

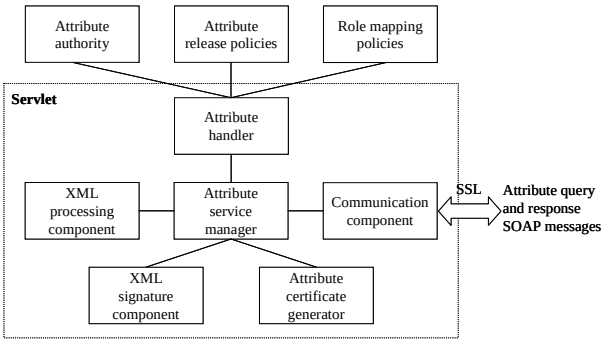

Figure 4. The attribute service structure

- *Attribute service manager* is the centre of the AttrS. It performs three primary functions. The first is asking the attribute handler to get a user's attributes from attribute authority or sending a new request to next AttrS. The second is passing one user's attributes to the attribute certificate generator to issue a X.509 AC for the user. User's AC is put into the SOAP message body. The third is asking the XML signature component to sign or verify a SOAP message.
- *Attribute handler* is responsible for contacting with attribute authority to get a user's attributes. It may also additionally consult the attribute release policies in order to decide what attributes should be released to the attribute requester. Finally, it may do the attributes interpretation according to the attribute mapping policies that specify how to convert an organization's inner attributes into outer attributes.
- *XML processing component* is responsible for creating SOAP messages, e.g. adding XML signature description into a SOAP message.
- *XML signature component* is responsible for signing and verifying SOAP messages.
- *Attribute certificate generator* is responsible for issuing X.509 ACs.
- *Communication component* is responsible for communication with other entities, e.g. an ACE or another AttrS, to send attribute request messages or receive attribute response messages. The connection is made through mutual authentication over SSL.

Authentication and authorization sequences

There are four types of messages that are involved the information exchanges, i.e. *authentication request*, *handle response*, *attribute request* and *attribute response*. In a generic application scenario, i.e. two sites are involved, the CSACF acts as follows (step sequence relates to Figure 2):

- A user connects to an ACE-protected web site, and is redirected to the destination site AuthS for authentication. (step 1)
- The destination site AuthS finds the user's origin site and redirects him to his origin site AuthS with an authentication request message (not show in the Figure 2). (step 2)
- The origin site AuthS authenticates the user and creates a handle (gets from AttrS). This handle is encapsulated in a handle response message. (step 2 and step 3)
- This user is redirected back to the destination site AuthS with the handle response message. After impersonation check, he is redirected to the ACE. (step 4)
- The AEF sends an attribute request message to the destination site AttrS based on the user's handle. (step 5)
- The destination site AttrS sends a new attribute request message to the user's origin site AttrS (not show in the Figure 2). The origin site AttrS gets the user's attributes and encapsulates them in a X.509 AC, and sends it back to the requester via attribute response message. (step 6)
- Based on the user's access request and his attributes, the AEF submits a decision request to the ADF. (step 7)
- The ADF makes an access decision according to the access control policies. (step 8)
- The AEF enforces the decision made by ADF; either accesses the target on behalf of the user or refuses this request. (step 9)

Message examples

In the next two examples, the user's origin site is "University2.Science.Engineering.VRC", and the destination site is "University1.Science.Engineering.B2B". A simplified authentication request message example is depicted in Figure 5. The destination site's ACE starts an authentication request process. The ACE adds a "Target" element that describes the user's desired target URL into the message, and passes it to the local AuthS. The local AuthS interacts with user, and adds a "Domain" element that describes where this message need to be sent, what the response message receiver address is, and where the message is forwarded to, and then passes this message to the user's origin site's AuthS. Each AuthS will do the same work.

```
<?xml version="1.0" encoding="UTF-8" ?>
<AttributeHandleQuery>
  <Target>http://University1/b2b/Enquiry</Target>
  <Domain>
    <Local>University1.Science.Engineering.B2B</Local>
    <RequestTo>University2.Science.Engineering.VRC</RequestTo>
    <ResponseTo/>
    <Receiver>https://University1/b2b/AuthSResponse</Receiver>
  </Domain>
  <Domain>
    <Local>University2.Science.Engineering.VRC</Local>
    <RequestTo>University2.Science.Engineering.VRC</RequestTo>
    <ResponseTo>University1.Science.Engineering.B2B</ResponseTo>
    <Receiver>https://University2/vrc/AuthSResponse</Receiver>
  </Domain>
</AttributeHandleQuery>
```

Figure 5. Example of authentication request message

A simplified handle response message example is depicted in Figure 6. After receiving an authentication request message from destination site AuthS, the origin site AuthS starts an attribute query handle response process. The AuthS first interacts with the user in order to authenticate him, and then contacts the local AttrS to get an attribute query URL about the user. The user's general information is saved in element "Person". The attribute query URL is saved in element "AttributeServices". Each domain saves its attribute service information in its own element "AttributeService". The received authentication request message is also saved in element "AuthenticationRequest" so that the response message can be sent back along the required route until arrives the initial requester, i.e. the destination site's ACE.

```
<?xml version="1.0" encoding="UTF-8" ?>
<AttributeHandleResponse>
  <UserHandle>
    <Person>
      <UserID>10002</UserID>
      <Organization>University2.Science.Engineering.VRC</Organization>
    </Person>
    <AttributeServices>
      <AttributeService ID="University2.Science.Engineering.VRC">
        <URL>https://University2/vrc/AttributeService</URL>
        <SupAttributeAuthority />
      </AttributeService>
      <AttributeService ID="University1.Science.Engineering.B2B">
        <URL>https://University1/b2b/AttributeService</URL>
        <SupAttributeAuthority>University2.Science.Engineering.VRC
        </SupAttributeAuthority>
      </AttributeService>
    </AttributeServices>
  </UserHandle>
  <AuthenticationRequest>
    <Target>http://University1/b2b/Enquiry</Target>
    <Domain>
      … … …
    </Domain>
  </AuthenticationRequest>
</AttributeHandleResponse>
```

Figure 6. Example of handle response message

4. ROLE MAP MECHANISM

The main purpose of CSACF is to securely transfer users' attributes between their origin site and the destination site to be accessed. But simply transferring attributes is not enough, generally one organization cannot directly use the attributes (roles) defined in anther organization. To fix this problem we introduce role contexts which are used for grouping and managing roles. Roles have functionalities only in a certain role context. One role context may cover several organizations, and one organization may contain several role contexts. Transferring privileges held by roles between different role contexts needs role mapping mechanism to do the privileges interpretation.

The transformation is done through an interface that is called virtual role context. Like a normal RBAC, this virtual role context contains specific roles and role hierarchy. Virtual role context participants contractually agree to its legal role structure, and define the mapping from their inner role structure to the virtual role context's role structure. With the help of the intermediary layer, two organizations' role structures indirectly interact with each other. The changes in the participating organizations do not affect the intermediary context security infrastructure. Instead, these changes are confined to

modification of the mapping from one organization's role structure to the virtual role context's role structure. An organization links or unlinks to a virtual role context does not affect its security infrastructure.

5. SUPPORT VIRTUAL ORGANIZATION

Virtual Organization (VO) is a popular concept for modeling collaboration application environment. A VO is a dynamic collection of resources and users unified by a common goal and potentially spanning multiple administrative domains [12]. VO may apply some common policy about how its users access the resources assigned to the VO, but each organization will typically retain ultimate control over the policies that govern access to its resources. The dynamic and multi-institutional nature of these environments introduces challenging security issues that demand new technical approaches. Since VO resources and users are located within multiple organizations, a key problem associated with the formation and operation of distributed virtual organizations is how to authenticate the users and enforce the community policies. We will describe how these issues are addressed by our CSACF by an example.

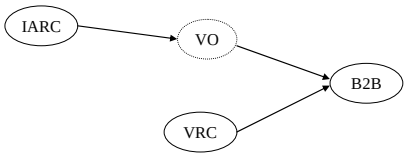

Figure 7. An example of virtual organization

The structure of the example is depicted in Figure 7. B2B, VRC and IARC are three independent research centers. VO is a virtual organization. B2B is a resources provider. VRC and IARC are identity providers. VRC and VO register in B2B, and their users can be authenticated and authorized through the way that we discussed early, and then visit the resources in B2B. IARC does not register in B2B, but it registers in VO. Because our framework supports transitivity, so the B2B can authenticate a user in IARC and obtain his attributes through the VO. The VO can enforce some common policies to the user's attributes when it forwards the message to its destination site.

The RBAC of a VO runs in a virtual role context, its role structure is contractually agreed by the participants. Every organization that registers in the VO will define a role mapping mechanism, through which its inner role structure interacts with the VO's role structure, and further interacts with other organizations' role structures. As any changes in an organization is confined to modification of the mapping from its inner role structure to the VO's role structure. Adding or removing an organization from the VO does not affect the VO and other participants' security infrastructure. So this mechanism provides real flexibility to the virtual organization management.

6. IMPLEMENTATION

We have developed a prototype based on our proposed CSACF framework. The implementation of the prototype uses the following software:

- Apache Web Server [14] provides both static HTML pages and the servlets' dynamic content.
- Jakarta Tomcat servlet container [15] is used for running the servlets that implement the ACE and CSC.
- SOAP with Attachments API for Java (SAAJ) [16] is used for implementing the communication among AuthS, AttrS and ACE.
- IBM XML Security Suite [17] provides the security features such as digital signature, encryption for XML documents.
- IAIK-JCE [18] provides the functions that relate to PKI and PMI, e.g. creates X.509 attribute certificates.
- MySQL [19] is used as a database server.
- OpenLDAP [20] is used as an attribute certificates repository.

All the software either are open-source software or free download for evaluation. Currently, this prototype is used for demonstrating the system's feasibility.

7. RELATED WORKS

There are three important related works. One is the Internet2 Shibboleth project [10], the Community Authorization Service (CAS) [11, 12] and the Virtual Organization Membership Service (VOMS) [13].

Shibboleth

Shibboleth is a cross-institutional authentication and authorization service for access control to Web-accessed resources. It is being specified by the Internet2 middleware architecture committee. It provides a secure framework for one organization to transmit attributes about a web-browsing individual across security domains to another institution.

There are two major differences between Shibboleth and CSACF. The first difference is that in Shibboleth the authentication and authorization service only happen between two institutions. Whereas the CSACF supports transitivity, it can pass the authentication or authorization service to next entity until a user is authenticated or authorized. The second difference is that Shibboleth does not have access control system. On the contrary, our CSACF includes an access control engine that supports RBAC. The CSACF supports transitivity is because our primary target is developing a mechanism for exchanging authentication and authorization information among big organizations that normally have hierarchical structures, e.g. governments. Through transitivity sub-organizations can be easily added or removed from the system. This issue is not addressed by Shibboleth.

CAS and VOMS

The Community Authorization Service (CAS) is developed by the Globus project for Grid environments. This authorization model allows a resources site to grant a community to access its resources, and the authorization server of that community to grant privileges its members. The Virtual Organization Membership Service (VOMS) is another solution to authentication in a GSI-enable Grid. The VOMS server is run by a virtual organization and supplies authorization information about its own members. CAS and VOMS are similar architecturally. Both of them issue policy assertions to a user. The user then presents these assertions to a resource for the purpose of obtaining VO-issued rights. The primary difference between the two systems is the level of granularity at which they operate. The VOMS assertions contain a list of role or

group memberships held by the user. CAS assertions provide the rights directly and do not need interpretation by the resource.

There are two major differences between CSACF and CAS or VOMS. In CAS and VOMS the users' authorization information is managed in a central server. Whereas in CSACF, the users' authorization information is managed in the users' origin sites, but through the transitivity the CSACF can perform common policies on the users. The second difference is that in CAS and VOMS users' authentication is done at resource provider sites; in CSACF users are authenticated at their origin sites. The major benefit of CSACF is the resource provider does not have to maintain partner's users. When the number of users gets large, the burden of managing identities for foreign users can become high. On the other hand, in CVS and VOMS the authentication system must be PKI based; in CSACF there is no such requirement, the identity provider decides what kind of authentication system should be used.

8. CONCLUSIONS AND FUTURE WORKS

With the CSACF we can get two major benefits. The first is independent organizations can share their resources without the burden of managing the users who belong to other organizations. They only need to manage the trust relationships with other organizations. The second is the CSACF supports virtual organization management. Independent organizations can be freely added or removed from the VO, and these modifications do not influence the security infrastructure of the VO or other participating organizations.

In our prototype the role mapping mechanism and the virtual organization management are still in the primary phase. The future works will be mainly around these two issues. Some GUI management tools will also be developed.

9. REFERENCES

[1] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman, **Role based access control models**, IEEE Computer, 29 February 1996.

[2] David F. Ferraiolo, R.S. Sandhu, Serban Gavrila, D. Richard Kuhn and Ramaswamy Chandramouli, **Proposed NIST Standard for Role-Based Access Control**, ACM Transactions on Information and Systems Security (TISSEC), Volume 4, Number 3, August 2001.

[3] ITU-T Rec. X.509 ISO/IEC 9594-8, **The Directory: Public-key and Attribute Certificate Frameworks**, May 3, 2001.

[4] R. Housley, W. Ford, W. Polk, D. Solo, **Internet X.509 Public Key Infrastructure Certificate and CRL Profile**, January 1999 , http://www.ietf.org/rfc/rfc2459.txt.

[5] S. Farrell, R. Housley, **An Internet Attribute Certificate Profile for Authorization**, April 2002, http://www.ietf.org/rfc/rfc3281.txt.

[6] The World Wide Web Consortium (W3C), **XML-Signature Syntax and Processing W3C Recommendation**, 12 February 2002, http://www.w3.org/TR/xmldsig-core/.

[7] The World Wide Web Consortium (W3C), **XML Encryption Syntax and Processing W3C Recommendation**, 10 December 2002, http://www.w3.org/TR/xmlenc-core/.

[8] ITU-T Rec. X.812(1995)|ISO/IEC 10181-3:1996, **Security frameworks in open systems: Access control framework**.

[9] W. Zhou, C. Meinel, **Implement role based access control with attribute certificates**, The 6th International Conference on Advanced Communication Technology (ICACT2004), Volume 1, P. 536-541, Korea, February 2004.

[10] M. Erdos, S. Cantor, **Shibboleth-Architecture DRAFT v05**, http://shibboleth.internet2.edu/.

[11] L. Pearlman, V. Welch, I. Foster, C. Kesselman, S. Tuecke, **A Community Authorization Service for Group Collaboration**, IEEE 3rd International Workshop on Policies for Distributed Systems and Networks, 2002.

[12] Ian Foster, Carl Kesselman, Laura Pearlman, Steven Tuecke, Von Welch, **The Community Authorization Service: Status and future**, CHEP 03, http://www.globus.org/security/CAS/GT3/.

[13] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Gianoli, F. Spataro, **VOMS: an Authorization System for Virtual Organizations**, Presented at the 1st European Across Grids Conference, Santiago de Compostela, February 13-14, 2003.

[14] **Apache Web Server**, http://www.apache.org/.

[15] **Jakarta Tomcat servlet container**, http://jakarta.apache.org/tomcat/.

[16] **SOAP with Attachments API for Java (SAAJ)**, http://java.sun.com/xml/saaj/.

[17] **IBM alphaworks, XML Security Suite**, http://www.alphaworks.ibm.com/tech/xmlsecuritysuite.

[18] **IAIK Java Cryptography Extension (IAIK-JCE)**, http://jce.iaik.tugraz.at/index.php.

[19] **MySQL, the open source SQL database server**, http://www.mysql.com/.

[20] **OpenLDAP, the open source Lightweight Directory Access Protocol (LDAP)**, http://www.openldap.org/.